

0380959  
247087US2SRD

日 本 国 特 許 庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日            2 0 0 3 年 1 1 月 1 9 日  
Date of Application:

出 願 番 号            特 願 2 0 0 3 - 3 8 9 7 1 0  
Application Number:  
[ST. 10/C]:            [ J P 2 0 0 3 - 3 8 9 7 1 0 ]

出 願 人            株式会社東芝  
Applicant(s):

2 0 0 3 年 1 2 月    9 日


特許庁長官  
Commissioner,  
Japan Patent Office

今 井 康



出証番号    出証特 2 0 0 3 - 3 1 0 1 8 2 3

【書類名】 特許願  
【整理番号】 A000305705  
【提出日】 平成15年11月19日  
【あて先】 特許庁長官 殿  
【国際特許分類】 G06F 15/00  
【発明者】  
    【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝研究開発セ  
                                ンター内  
    【氏名】 近藤 浩一  
【発明者】  
    【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝研究開発セ  
                                ンター内  
    【氏名】 吉田 充伸  
【特許出願人】  
    【識別番号】 000003078  
    【氏名又は名称】 株式会社 東芝  
【代理人】  
    【識別番号】 100058479  
    【弁理士】  
    【氏名又は名称】 鈴江 武彦  
    【電話番号】 03-3502-3181  
【選任した代理人】  
    【識別番号】 100091351  
    【弁理士】  
    【氏名又は名称】 河野 哲  
【選任した代理人】  
    【識別番号】 100088683  
    【弁理士】  
    【氏名又は名称】 中村 誠  
【選任した代理人】  
    【識別番号】 100108855  
    【弁理士】  
    【氏名又は名称】 蔵田 昌俊  
【選任した代理人】  
    【識別番号】 100084618  
    【弁理士】  
    【氏名又は名称】 村松 貞男  
【選任した代理人】  
    【識別番号】 100092196  
    【弁理士】  
    【氏名又は名称】 橋本 良郎  
【先の出願に基づく優先権主張】  
    【出願番号】 特願2002-376209  
    【出願日】 平成14年12月26日  
【手数料の表示】  
    【予納台帳番号】 011567  
    【納付金額】 21,000円  
【提出物件の目録】  
    【物件名】 特許請求の範囲 1  
    【物件名】 明細書 1



【物件名】 図面 1  
【物件名】 要約書 1  
【包括委任状番号】 9705037

**【書類名】 特許請求の範囲****【請求項 1】**

複数の変数を有する連続系方程式から構成されるダイナミクスモデルを用いて機構の挙動をシミュレーションするダイナミクスシミュレーションを実行する第 1 のシミュレータと、複数の機構要素から構成される 3 次元機構モデルを用いて前記機構の 3 次元空間内での幾何学的な動作をシミュレーションするキネマティックスシミュレーションを実行する第 2 のシミュレータとを連携させる機構シミュレーション方法であって、

前記第 1 のシミュレータが前記ダイナミクスシミュレーションを実行することにより前記連続系方程式のいずれかの変数の値を計算するステップと、

前記複数の変数と前記複数の機構要素との対応を表すテーブルを参照し、前記値が計算された変数に対応付けられる機構要素を特定するステップと、

計算された前記変数の値と特定された前記機構要素とを表す情報を前記第 2 のシミュレータに送信するステップと、

前記情報に基づいて、前記第 2 のシミュレータが前記キネマティックスシミュレーションを実行するステップとを具備する機構シミュレーション方法。

**【請求項 2】**

前記ダイナミクスモデルは、連続系モデルと状態遷移モデルとからなるハイブリッドモデルを含み、前記ダイナミクスシミュレーションはハイブリッドシミュレーションを含む請求項 1 に記載の方法。

**【請求項 3】**

前記状態遷移モデルには前記機構を制御する機構制御ソフトウェアを含む外部からの制御信号が入力される請求項 2 に記載の方法。

**【請求項 4】**

前記機構要素はアクチュエータの回転角ないしは移動量を含む請求項 1 に記載の方法。

**【請求項 5】**

請求項 1 に記載の方法であって、前記複数の変数と前記複数の機構要素との対応を表すテーブルを作成するステップをさらに具備し、該ステップは以下を含む：

前記複数の変数を表すダイナミクスモデルのデータを読み込むステップ；

前記複数の機構要素を表す 3 次元機構モデルのデータを読み込むステップ；

前記ダイナミクスモデルのデータから、前記複数の機構要素のいずれかに対応付けることが可能な変数を抽出するステップ；

前記 3 次元機構モデルのデータから、前記複数の変数のいずれかに対応付けることが可能な機構要素を抽出するステップ；および

抽出された変数のいずれかと抽出された機構要素のいずれかとの組み合わせをユーザに選択させるステップ。

**【請求項 6】**

請求項 5 に記載の方法であって、前記組み合わせをユーザに選択させるステップは以下を含む：

抽出された各々の変数が属するクラスのいずれかを前記ユーザに選択させる第 1 のステップ；

前記第 1 のステップにより選択されたクラスに属するメンバー変数を前記ユーザに選択させる第 2 のステップ。

**【請求項 7】**

前記ダイナミクスモデルのデータは、ハイブリッドモデリング言語の記述データを含む請求項 5 に記載の方法。

**【請求項 8】**

生成されたテーブルをファイルに格納するステップをさらに具備する請求項 5 に記載の方法。

**【請求項 9】**

複数の変数を有する連続系方程式から構成されるダイナミクスモデルを用いて機構の挙動

をシミュレーションするダイナミクスシミュレーションを実行する第1のシミュレータと、複数の機構要素から構成される3次元機構モデルを用いて前記機構の3次元空間内での幾何学的な動作をシミュレーションするキネマティックスシミュレーションを実行する第2のシミュレータとを連携させる機構シミュレーションプログラムであって、

前記第1のシミュレータが前記ダイナミクスシミュレーションを実行することにより前記連続系方程式のいずれかの変数の値を計算する手順と、

前記複数の変数と前記複数の機構要素との対応を表すテーブルを参照し、前記値が計算された変数に対応付けられる機構要素を特定する手順と、

計算された前記変数の値と特定された前記機構要素とを表す情報を前記第2のシミュレータに送信する手順と、

前記情報に基づいて、前記第2のシミュレータが前記キネマティックスシミュレーションを実行する手順とをコンピュータに実行させる機構シミュレーションプログラム。

【請求項10】

前記ダイナミクスモデルは、連続系モデルと状態遷移モデルとからなるハイブリッドモデルを含み、前記ダイナミクスシミュレーションはハイブリッドシミュレーションを含む請求項9に記載のプログラム。

【請求項11】

前記状態遷移モデルには前記機構を制御する機構制御ソフトウェアを含む外部からの制御信号が入力される請求項10に記載のプログラム。

【請求項12】

前記機構要素はアクチュエータの回転角ないしは移動量を含む請求項9に記載のプログラム。

【請求項13】

請求項9に記載のプログラムであって、前記複数の変数と前記複数の機構要素との対応を表すテーブルを作成する手順をさらに具備し、該手順は以下を含む：

前記複数の変数を表すダイナミクスモデルのデータを読み込む手順；

前記複数の機構要素を表す3次元機構モデルのデータを読み込む手順；

前記ダイナミクスモデルのデータから、前記複数の機構要素のいずれかに対応付けることが可能な変数を抽出する手順；

前記3次元機構モデルのデータから、前記複数の変数のいずれかに対応付けることが可能な機構要素を抽出する手順；および

抽出された変数のいずれかと抽出された機構要素のいずれかとの組み合わせをユーザに選択させる手順。

【請求項14】

請求項13に記載のプログラムであって、前記組み合わせをユーザに選択させる手順は以下を含む：

抽出された各々の変数が属するクラスのいずれかを前記ユーザに選択させる第1の手順；

前記第1の手順により選択されたクラスに属するメンバー変数を前記ユーザに選択させる第2の手順。

【請求項15】

前記ダイナミクスモデルのデータはハイブリッドモデリング言語の記述データを含む請求項13に記載のプログラム。

【請求項16】

生成されたテーブルをファイルに格納する手順をさらに具備する請求項13に記載のプログラム。

【書類名】明細書

【発明の名称】機構シミュレーション方法および機構シミュレーションプログラム

【技術分野】

【0001】

本発明は、コンピュータを用いて機構の挙動をシミュレーションする方法およびプログラムに関し、特に微分方程式ないしは代数方程式からなる連続系方程式を用いた機構の時間軸に沿った挙動に関するダイナミクスシミュレーションと、3次元の形状データを含む3次元機構モデルを用いたキネマティックスシミュレーションとを併用するものである。

【背景技術】

【0002】

現在、コンピュータを用いて機械やプラント等の連続系の時間軸上での挙動のシミュレーションを行う際には、対象を微分方程式などでモデル化し、数値積分などの手法を用いて解を求めることが行われている。さらに、複雑な系の挙動を表現するためには、ハイブリッドモデリングと呼ばれる手法が使われることもある。ハイブリッドモデルを用いたシミュレーションは、通常の変微分方程式によるシミュレーションをさらに高度化したものであり、「ハイブリッドシミュレーション」と呼ばれている。このようなシミュレーション挙動をするシステムを「ハイブリッドシステム」と呼ぶこともある。シミュレーションの目的で作成されるハイブリッドモデルは、概念的には常微分方程式や代数方程式を連立させた連立方程式によって表現される連続系モデルと、イベント発生に伴う状態遷移を表現するための状態遷移モデルとを組み合わせたモデルである。ハイブリッドモデルによれば、連続系モデルで表現される状態が外部からのイベントなどにより瞬時に切り替わるシステムを表現することができ、より高度なモデル表現が可能である。

【0003】

ハイブリッドモデルを記述するための言語として、米国ゼロックス社（商標）のパロアルト研究所にて創作されたHCC（Hybrid Concurrent Constraint Programming）と呼ばれる言語がある（下記非特許文献1参照）。HCCは発展途上にあり、現在も米国NASAのエイムズ研究所において研究が進められている。HCCは制約処理プログラミング（コンストレイントプログラミング）と呼ばれる技術の一種であり、連続系モデルを表現する常微分方程式や代数方程式を制約として扱い、これら方程式をそのまま順不同で記述することができる。このような制約記述に、状態遷移を制御する記述を付加してHCC言語のハイブリッドモデルは完成される。HCCによれば、方程式をそのまま制約として羅列する（プログラミングする）ことができ、複雑なモデルを記述することが可能である。

【0004】

このようにハイブリッドモデルの技術を用いれば、系の特性を常微分方程式などでモデル表現し、初期状態から時間の推移に従ってどのような挙動を示すかをシミュレーションすることができる。このような微分方程式などの方程式によるモデル化は外乱による過渡応答や振動など動的な挙動を表現できるため、ダイナミクスシミュレーション（動力学シミュレーション）とも呼ばれる。

【0005】

微分方程式などで表現可能な対象や現象を的確にモデル化できるハイブリッドモデルの技術の応用例として、ソフトウェアにより機構が制御されるメカトロニクス機器の機構シミュレーションがある。かかる機構シミュレーションによれば、機構の実機が存在しない状況においても、当該機構を制御する制御ソフトウェアのプロトタイピング、テスト、あるいはデバッグなどが行えるようになる。

【0006】

一方、機構の3次元的な形状データや、部品同士の関節などによる接続関係の情報などを用いて、機構の3次元空間内での幾何学的な動作のシミュレーションを行う技術が知られている。3次元CADの技術や機構シミュレーションソフトウェアがこれに相当し、例えば、特開2001-222572公報（下記特許文献1）に記載の技術も参考にすることができる。このようなソフトウェアは振動特性のような時間軸に沿った系の挙動のシミュレーションを行うことができる。

ュレーションはできないが、複雑な位置関係などを扱うことができ、キネマティクスシミュレーション（機構学シミュレーション）と呼ばれることもある。

#### 【0007】

特開 2001-222572 に記載の技術は、ソフトウェアにより機構が制御されるメカトロニクス機器の機構シミュレーションを、キネマティクスシミュレーションおよび限定的かつすでにパッケージソフトウェアに組み込み済みの限られたダイナミクスの範囲において、実現した公知例である。

#### 【0008】

機構シミュレーションソフトウェアの中には、ダイナミクスシミュレーションの機能を取り入れ、機構学と動力学の両方を扱えるようにしたものも存在する。しかし、このようなソフトウェアは、前記 HCC で示したようなプログラミング言語的に、複雑な系のモデル化を柔軟に行えるダイナミクスシミュレーションには対応していない。このため、ユーザがシミュレーションの目的に応じて最適なモデル化手法を選び、それに基づいて作成したモデル記述を用いてキネマティクスシミュレーションとダイナミクスシミュレーションを並立させて実行することができなかった。さらに、より複雑なハイブリッドモデリング技術を用いる場合には、キネマティクスシミュレーションとダイナミクスシミュレーションを連携させることは困難であった。

【非特許文献 1】特開 2001-222572 公報

【非特許文献 1】インターネット<URL: [www2.parc.com/spl/projects/mbc/publications.html#cclanguages](http://www2.parc.com/spl/projects/mbc/publications.html#cclanguages)>

#### 【発明の開示】

#### 【発明が解決しようとする課題】

#### 【0009】

本発明はかかる事情を考慮してなされたものであり、微分方程式などの方程式記述によるダイナミクスモデルやハイブリッドモデルを用いた複雑なダイナミクスモデルと 3 次元形状データを含むキネマティクスモデルを容易に連携させることを可能とすることを目的とする。

#### 【0010】

また、これにより、ダイナミクスシミュレーションとキネマティクスシミュレーションを並立した高度なシミュレーションを、ソフトウェアにより機構が制御されるメカトロニクス機器の実機レスシミュレーションに適用可能にすることを目的にする。

#### 【課題を解決するための手段】

#### 【0011】

本発明の一観点に係る機構シミュレーション方法は、複数の変数を有する連続系方程式から構成されるダイナミクスモデルを用いて機構の挙動をシミュレーションするダイナミクスシミュレーションを実行する第 1 のシミュレータと、複数の機構要素から構成される 3 次元機構モデルを用いて前記機構の 3 次元空間内での幾何学的な動作をシミュレーションするキネマティクスシミュレーションを実行する第 2 のシミュレータとを連携させる機構シミュレーション方法である。この機構シミュレーション方法は、前記第 1 のシミュレータが前記ダイナミクスシミュレーションを実行することにより前記連続系方程式のいずれかの変数の値を計算するステップと、前記複数の変数と前記複数の機構要素との対応を表すテーブルを参照し、前記値が計算された変数に対応付けられる機構要素を特定するステップと、計算された前記変数の値と特定された前記機構要素とを表す情報を前記第 2 のシミュレータに送信するステップと、前記情報に基づいて、前記第 2 のシミュレータが前記キネマティクスシミュレーションを実行するステップとを具備する。

#### 【発明の効果】

#### 【0012】

本発明によれば、ハイブリッドモデルを用いるようなダイナミクスシミュレーションと 3 次元キネマティクスシミュレーションを簡便かつ効果的に連携させることが可能となり、かつダイナミクスシミュレーションは、ハイブリッドモデル記述言語のようなプログラ

ミング言語により自由に記述することが可能となる。これにより、複雑な機構系を簡便かつ正確にモデル化でき、該機構系を制御する制御ソフトウェアとの連携シミュレーションにも好適な機構シミュレーション方法およびプログラムを提供できる。

【発明を実施するための最良の形態】

【0013】

以下、図面を参照しながら本発明の実施形態を説明する。なお、以下では、ダイナミクスシミュレータとしてハイブリッドモデリング技術を採用し、かつ、機構制御ソフトウェアないしは機構制御ソフトウェアのシミュレータと連携して動作する最も複雑な実施形態の1つの態様をもとに説明する。

【0014】

<第1の実施形態>

図1は、本発明の第1実施形態に係る機構シミュレータの概略構成を示すブロック図であり、図2は、かかる機構シミュレータが機構制御ソフトウェアないしは機構制御ソフトウェアのシミュレータと連携する場合の構成を示している。

【0015】

機構シミュレータはダイナミクスシミュレーション部101およびキネマティクスシミュレーション部103および変数・機構要素対応テーブル105から構成され、機構制御ソフトウェアないしは機構制御ソフトウェアのシミュレータ108と制御信号109をやり取りする。なお、本実施形態は一般的なコンピュータ107を用いて構成することができ、その基本的なハードウェア構成として、図示しないが中央演算ユニット（CPU）、メモリ、外部記録装置、通信インターフェース（I/F）、および表示装置やキーボード、マウス等の入力装置を備える。また、これらのハードウェアを制御するためのオペレーティングシステム（OS）を備える。また、本発明の実施形態に係る機構シミュレータは、このようなオペレーティングシステム上で動作するアプリケーションソフトウェアとして実装することができる。

【0016】

シミュレーション対象である機構の動力学的なモデル記述は、後で詳細に説明するハイブリッドモデル記述言語により記述され、ファイル（ハイブリッドモデル記述）102としてあらかじめ記憶されている。一方、3次元形状情報を含む機構モデルデータも所定のフォーマットに従いファイル（3次元機構モデル）104としてあらかじめ保存されている。これらのデータはシミュレーション開始時に読み込まれる。この段階では、ダイナミクスシミュレーション部101およびキネマティクスシミュレーション部103がそれぞれ単独に動作することが可能である。次に、あらかじめ保存されていたダイナミクスモデル記述に現れる変数と3次元機構モデルの機構要素との対応関係データ106を読み込む、ないしはPCのキーボードなど入力デバイスより直接データを入力することにより、変数・機構要素対応テーブル105のデータがセットされる。これにより、ダイナミクスシミュレーション部101およびキネマティクスシミュレーション部103が連携して動作する準備が整う。

【0017】

実際のシミュレーションにおいては、モータなどアクチュエータへの加速コマンドや加速レートのパラメータ設定などの情報が、機構制御ソフトウェアないしは機構制御ソフトウェアのシミュレータ108から送信され、それに基づきハイブリッドモデルとして記述された適切な状態遷移と方程式に基づく数値積分による系の挙動が計算される。ハイブリッドモデルにより、どのようにダイナミクスが表現され、状態遷移が実現できるかについては、後で詳細に説明する。

【0018】

実際の機構制御系においては、機構制御ソフトウェアからのコマンドやパラメータは、一定の時間サイクルごとに指定されたI/Oポートを読み込みにくポーリング（Polling）処理によって取得される。そのためシミュレーションにおいても、これと同じ時間間隔ごとにシミュレーションをすすめていく。



## 【0019】

すなわち、図3に示されるフローチャートのように、ハイブリッドモデルの読み込みステップ501および3次元機構モデル読み込みステップ502を実行し（501と502の順序は入れ替わった実施形態としても全く問題はない）、シミュレーションを開始する。前記一定のサイクルごとに制御信号を読み込みに行く制御信号受信ステップ503を行い、それに基づく適切な状態遷移と方程式に基づく数値積分による系の挙動が計算されるダイナミクスシミュレーションステップ504が実行される。これにより、各変数の値が計算されるので、変数と機構要素の対応関係をチェックしキネマティクスシミュレーション部に値を送信するステップ505が実行され、その値に基づきキネマティクスシミュレーションがステップ506で実行される。機構制御系は、機構からセンサー情報を受け取って、新たな制御コマンドの必要性を判断するので、ダイナミクスシミュレータ101ないしはキネマティクスシミュレータ103から得られる主にセンサーデータを機構制御系に送信するステップ507が実行される。このあと、あらかじめ指定された時間だけシミュレーションが進んだかどうかを判断するステップ508を行い、まだ実行途中であれば次の時間に進めるために、データの記録などの処理を行うステップ509が実行され、ふたたび制御信号の受信ステップ503にもどる。

## 【0020】

このような本実施形態に係るダイナミクスシミュレーションのためのハイブリッドモデル記述102の詳細を、具体例を挙げて説明する。

## 【0021】

図4および図5は、具体例に係るハイブリッドモデルの記述対象である機構を示す図である。この機構は、バルブ301、バネ303、およびピストン302を備えた簡素な構造を有するシリンダ装置である。

## 【0022】

バルブ301は外部からの指令（イベント）に応じて開閉動作する。これによりシリンダ装置内の空気の流れを図2のように右側に変更するイベントを以下、「Right」と呼び、空気の流れを図3のように左側に変更するイベントを「Left」と呼ぶ。図4は、バルブ301にRightのイベントが与えられた状態を示しており、ピストン302には紙面左向きへの力が作用している。この状態を示す運動方程式はシリンダ装置の下部に示してあるように、「 $-F = m \cdot x$ 」である。これに対し図5は、バルブ301にLeftのイベントが与えられた状態を示しており、空気の流れの向きが変わり、運動方程式は同図のように「 $F = m \cdot x$ 」に変化している。

## 【0023】

図6は、このような状態変化とそれぞれの状態に対応する運動方程式を状態遷移図として表現したものである。ハイブリッドモデルは、この図6に示されるような状態遷移と、各状態の記述が微分方程式や代数方程式で表現されるものを指す。図6によれば、状態が2つあって、かかる2つの状態間に状態遷移が存在することがわかる。

## 【0024】

図7は、図6の状態遷移図をもとに具体的なハイブリッドモデル内容をHCC（Hybrid Concurrent Constraint Programming）言語で記述したプログラムの一例を示す図である。図7において、（ソース）プログラムの論理行番号を仮にL1～L8とする。L3、L4、およびL8は、この機構の初期状態やバルブ操作タイミングなどの運転条件の記述に相当し、L5およびL6は、図6に示した状態遷移の表現記述である。

## 【0025】

HCCでは、運動方程式は、図から分かるようにプログラム内でそのまま記述することができる。また、それぞれの状態へ遷移する条件は、「always if」に続いて記述し、また、それぞれの状態から遷移していく条件は「watching」に続けて記述すればよい。

## 【0026】

なお、HCCでは、プログラムの記述の順序（図7における論理行番号L1→L8の順序）にそって実行されるわけではない。HCCでは、個別のプログラム記述のうち、シミ

ュレーションを実行する時間軸に沿って成立するものが探索され、実行される。すなわち、論理行番号  $L1 \rightarrow L8$  の順序は、実行順序とは関係がない。たとえば、シミュレーションを開始した時点では、 $L3$  および  $L8$  のみが有効である。ここで、イベント  $Right (ev1)$  が  $L3$  により発生するため、 $L6$  の前提条件である  $Right$  が有効となり、 $L6$  に記述されている運動方程式  $eq2$  が有効になる。つまり、図4の左側の状態からシミュレーションが実行されることになる。

#### 【0027】

さらに、時間が50になると  $L4$  が有効となり、イベント  $Left (ev2)$  が発生し、 $L6$  の遷移条件（「watching」以下、すなわち  $Left$ ）が有効となって、 $L6$  の運動方程式  $eq2$  が無効となる。これに代わって、 $L5$  の前提条件が有効となり、運動方程式  $eq1$  が有効となる。

#### 【0028】

なお、以上のようなプログラム例は外部からのイベント（ $ev3$ ,  $ev4$ ）によって状態が遷移する場合を記述したものであったが、勿論、内部の状況によって状態を変化させてもよい。たとえば、図4においてバルブ301が切り替えられない場合には、移動するピストン302がバネ303に接触し、該バネ303からの反力を受けるようになる。すなわちピストン302の位置に関して、外部からのイベントが無い場合でも状態遷移が起こる場合が存在する。このような場合は、例えば  $x$  が正であるかどうかといった内部変数の評価式（不等式）による評価結果に基づいて状態遷移の必要性を判断できる。

#### 【0029】

一般的に、ハイブリッドモデルは、常微分方程式や代数方程式をあるいはこれらを連立させた連立方程式（連続系方程式）によって表現される連続系モデルと、イベント発生に伴う状態遷移を表現するための状態遷移モデルとを組み合わせたモデルである。ハイブリッドモデルによれば、連続系モデルで表現される状態が外部からのイベントなどにより瞬時に切り替わるシステムを表現することができる。

#### 【0030】

次に、図10から図12を用いて、3次元機構モデル104およびキネマティクスシミュレーション部103の具体的な実施形態を説明する。

#### 【0031】

3次元機構モデル104では、図10に示すように個々の部品の形状が、その構成要素である面や稜線、頂点などの要素の集合として表現される。これら面や稜線、頂点などは部品の形状の境界を表現しているため、境界表現の形状モデルと呼ばれ、3次元のCADシステムなどで広く用いられている。このように表現された個々の部品の関係は、さらに部品の部分形状間の関係として表現される。この例では、部品401、部品402、部品403の3次元形状があり、部品401の部分形状である平面404と部品402の部分形状である平面406が一致し、同様に円筒面405と円筒面407が同軸であるという関係が定義されている。同様に部品402と部品403については、部分形状である平面408と平面410の一致関係、および平面409と平面411との一致関係が定義されている。

#### 【0032】

図11は、図10で定義された関係が計算機メモリなどの記憶手段にどのような形式で記憶されるかをグラフの形式で表現したものである。3次元形状の情報および部分形状の情報の他に、平面同士の一致関係412、414、415および円筒同士の同軸関係413がデータとして存在する。このような情報から部品401、402、403の相対的位置関係を計算する機能は幾何拘束処理ライブラリとして既存のソフトウェアが提供されている。具体的には、それぞれの部品には、部品特有のローカル座標系が設定されており、この座標系と空間内に固定されたワールド座標系との変換マトリックスという形でこの位置関係が表現される。すなわち、図11に示されている一致などの関係から、部品401と部品402、部品403の位置を表現する変換マトリックスが幾何拘束処理ライブラリなどにより自動的に計算される。

## 【0033】

図12は、部分形状の間で定義された一致関係、すなわち部分形状間の拘束関係をすべて満足するように算出された部品間の位置関係の例を示す。すなわち、幾何拘束処理ライブラリが自動算出した部品の変換マトリックスを部品の形状データに作用させて得られた部品の位置を示している。このように組み立てられた部品には、416で示される回転自由度と417で示される並進自由度が存在する。実際に機構として作用させるためには、これらにモータなどのアクチュエータを装着し、外部の信号に基づいて駆動することが行われる。そこで、回転自由度416に装着されるアクチュエータにJoint1という名前を付け、並進自由度417に装着されるアクチュエータにSlide1という名前を付ける。キネマティクスシミュレーション部103はアクチュエータの名前とその値を指定すると、図12のような3次元グラフィックスのアニメーションにて機構の動きを確認することなどが可能になる。また、図12に示されている距離418を計算する機能などはキネマティクスシミュレーション部103に実装されており、部品403が溝の端まで到達したかどうかを検知するリミットスイッチセンサーの機能をこの距離の値の判定によってシミュレートすることができる。このセンサーの名前をSwitch1とすると、キネマティクスシミュレーション部103はSwitch1という名前でセンサー状態を問い合わせると、そのときのセンサーの状態を得ることができる。

## 【0034】

今、アクチュエータSlide1は、実際には図4のシリンダ装置に取り付けられているとする。この場合、図7のダイナミクスシミュレータの記述における変数xが機構要素であるアクチュエータSlide1に対応する。さらにJoint1に相当するモータの挙動がダイナミクスシミュレータの記述において変数yとなっているとすると、変数・機構要素対応テーブル105は具体的に図13に示すようになる。

## 【0035】

このような例においては、図3のフローチャートにおいて、図7のダイナミクスモデル記述に基づきステップ504においてxおよびyの値の変化が計算される。次にステップ505において、変数・機構要素対応テーブル105である図12のテーブルが参照され、キネマティクスシミュレーション部103に対して、xの値とSlide1という名前の組でデータが送信され、同じくyの値とJoint1という名前の組でデータが送信される。ステップ506では、図12のような個々の部品の3次元的な位置が計算され、それに基づき距離418も算出される。距離418の値によりセンサーSwitch1の値もセットされる。ステップ507では、Switch1の値などが機構制御系へ送信される。

## 【0036】

## &lt;第2の実施形態&gt;

次に、本発明の第2の実施形態を説明する。第1の実施形態では公知のハイブリッドプログラミング言語であるHCCを用いて説明したが、ここでは本出願の発明者が開発したハイブリッドプログラミング言語であるDCML(Dynamics Constraint Modeling Language)に基づいて説明する。DCMLでは、HCCなど他のハイブリッドモデリング言語と同様に、連立常微分方程式で表現される連続系のシステムの初期値問題を解くことができる。すなわち、初期値を与えて、時間的にどのように系が変化するかをシミュレーションするダイナミクスシミュレーションを行うことができる。モデルの記述に当っては、常微分方程式ないしは代数方程式を制約の形式で宣言的に表現することができる。ただし、DCMLでは、処理の高速化などの目的から不等式は許されない。この点はHCCと異なる。DCMLはあるタイミングで連続系のシステム表現モデルを不連続的に切り替えることを許している。すなわち、イベントによってタイミングを指定し、そのタイミングにおいて常微分方程式ないしは代数方程式を制約の形式で表現されている制約の削除、追加、入れ替えを行うことができる。

## 【0037】

DCMLではオブジェクト指向の考え方に基づきクラスという形でモデル部品(ソフト

ウェア部品)を定義することができる。まずモデル部品の性質や挙動がクラス定義として記述され、個々のモデル部品の実体(インスタンス)はクラスのコンストラクタと呼ばれるメカニズムによって生成される。アクチュエータやセンサーなどがクラスとしてモデル化されていると、DCMLによるダイナミクスシミュレータ用のモデルは、これらクラスを活用してコンパクトに記述することができる。さらに制約を宣言的に順不同に記述できるという性質から、モデル部品のプログラムを単純に結合するのみで、全体のモデルを作成することができる。

#### 【0038】

以下、さらに例を用いてDCMLプログラムがどのように記述されるかを説明する。図14は不連続な変化を含まない単純なDCMLプログラムを示している。DCMLではC++に類似した文法を採っている。このプログラム例において、contは連続変数ないしは連続値を取る関数の宣言である。 $a=0$ ;  $b=0$ ; は時間 $t=0$ において連続変数の初期値を示す。制御文always は、対応する $b'=0.2$ ;  $a'=\sin(b)$ ; が対応する時間区間においてアクティブであることを示している。この例では、不連続的变化をおこすイベントは発生しないので、対応する時間区間は時間区間全体になり、すべての時間で、 $b'=0.2$ ;  $a'=\sin(b)$ ; はアクティブになる。最後の文 sample(a); では特別な組込み関数の呼び出ししており、指定された変数の時間変化を記録する。

#### 【0039】

DCMLにおいては、変数は連続変数とイベント変数があり、それぞれの宣言文にはcontとeventを用いる。連続系は連続変数を用いて表現される連続系拘束によって表現される。連続系拘束の表現の文法を形式的に表すと以下ようになる。

#### 【数1】

```

ContConst ::= Term RelOp Term
RelOp ::= =
Term ::= UVariableExpr | Term BinOp Term
UVariableExpr ::= VariableExpr | Constant |
                + UVariableExpr | - UVariableExpr
VariableExpr ::= Variable | VariableExpr.Variable |
                VariableExpr' | UnOp(VariableExpr) | (Term)
BinOp ::= + | - | * | / | ^
UnOp ::= sin | cos | tan | asin | acos | atan | sqrt | abs

```

#### 【0040】

ここで、Variableは変数名でありConstantは実数を表す。Variable' はVariableの時間微分となる。連続系はアクティブな連続系拘束の連立方程式としてモデル化される。連続系拘束のアクティブ化(有効化)と無効化は制御文で制御される。制御文の付いていない連続系拘束は時間0においてのみ有効になり、変数の初期値を指定する。連続系拘束をアクティブにしておくにはalways文を使う必要がある。例えば、以下のように用いる。

#### 【数2】

```

always a' = 1;
always { a' = 1; b' = 2; }

```

#### 【0041】

連続系システムの不連続変化は連続系拘束の有効化と無効化に対応する。このような制御は条件文の評価により実行される。DCMLでは条件文は2つの種類が用意されている。

#### 【0042】

最初の種類は、if( CondClaus1 ) A else Bの形式になる。文法を形式的に表すと以下のようになる。

## 【数 3】

```

CondClaus1 ::= CondTerm | CondTerm CondOp CondTerm
CondOp    ::= && | ||
CondTerm  ::= Term CondBinOp Term
CondBinOp ::= = | > | >= | < | <=

```

## 【0043】

例えば、以下の例は  $a > 1$  を評価する。その評価結果に基づき  $b$  の値が変化する。

## 【数 4】

```

if ( a > 1 ) b = 0;
else      b = 1;

```

## 【0044】

第2の種類は、条件文として1つの等式のみが認められるものである。これについて文法を形式的に表すと以下ようになる。

## 【数 5】

```

CondClaus2 ::= Term = Term

```

## 【0045】

この種類には2つの制御文がある。

## 【数 6】

```

when ( CondClause2 ) { A }
do { B } watching ( CondClause2 );

```

## 【0046】

上記when文は、CondClause2が成立するタイミングを探し、そのタイミングにおいて連続系拘束Aを有効化する。do-watching文は、まず連続系拘束Bを有効化し、次にCondClause2が成立するタイミングを探索し、連続系拘束Bを無効化する。

## 【0047】

DCMLでは、オブジェクト指向の考え方にに基づき、クラス定義を行うことができる。クラス定義はC++に類似した文法となっている。簡単なクラス定義の例を図15に示す。変数のスコープ（有効範囲）は、変数宣言のprivate文またはpublic文で定義される。パブリック変数はC++のような形式で外部からアクセスできる。例えば、図15中のパブリック変数m\_aはappl.m\_aという形式で外部から参照可能となる。クラスのコンストラクタDCML\_Class( cont a )は、この場合1つの引数cont aを持ち、この引数cont aはクラスの初期化に用いられる。def文はクラスの実体を規定しており、シミュレーション時に実行される部分である。この例では、状態遷移はm\_a' = 0が成立するタイミングにおいて発生する。

## 【0048】

図16は紙搬送系モデルの一例を示している。このモデルは説明のため実際のモデルよりも大幅に簡略化してある。モータや紙などのモデル部品はS1中に示されるようにあらかじめ用意されている。モータの初期化には、初期スピード、加速時間、停止タイミングを属性として指定できる。モータは、モータスタートコマンドを受け取ると指定された初期スピードで動きはじめ、加速する。その後、スピード一定モードに移り、指定されたタイミングで停止する。紙はベルトコンベアで搬送されることが想定されており、モータの回転各速度と紙の速度の関係がモデルに記述されている。図中S2に示されているのが、設計者によって追加されるべきプログラムである。最初の行はmotor1がpaper1の搬送系に取り付けられていることを示し、それらの速度が等しいことを表す。これに続く行の記述には、モータおよび紙の定義と制御シナリオが含まれる。この例では、モータスタートコ

マンドはtime = 10とtime = 70で発生する。実際のシミュレーションに際しては、あらかじめ定義された部分と設計者によって追加されるべきプログラムが結合され、ダイナミクスシミュレーションのためのシミュレーションモデルとなる。

#### 【0049】

さらに、図8を用いて、ハイブリッドモデル記述を用いたダイナミクスシミュレーションの詳細について説明する。図8は、図1および図2に示したダイナミクスシミュレーション部101の一実施形態を示したブロック図である。

#### 【0050】

本実施形態は、ハイブリッドモデル前処理部801と、ハイブリッドモデルシミュレーション実行部802とにより構成されている。ハイブリッドモデル記述102は、HCC言語で記述されたソースプログラムであって、本実施形態に係るハイブリッドモデル前処理部801への入力である。制御信号109はハイブリッドモデルシミュレーション実行部802への入力であり、この制御信号109は機構制御ソフトウェアもしくは機構制御ソフトウェアのシミュレータから与えられる。また、本実施形態に係る機構ハイブリッドモデルシミュレーション実行部802からの出力は、シミュレーション結果としての変数値の演算結果およびその時間履歴であり、変数値時間履歴記憶部805に対して出力される。

#### 【0051】

図8に示すように、ハイブリッドモデル前処理部801は、モデル方程式制御情報解析部811を備える。また、ハイブリッドモデルシミュレーション実行部802は、方程式構文解析部812、方程式データ記憶部814、連続系方程式切り替え部815、及び連続系シミュレーション部803とを備える。なお、本実施形態は一般的なコンピュータを用いて構成することができ、その基本的なハードウェア構成として、図示しないが中央演算ユニット(CPU)、メモリ、外部記録装置、通信インターフェース(I/F)、および表示装置やキーボード、マウス等の入力装置を備える。また、これらのハードウェアを制御するためのオペレーティングシステム(OS)を備える。また、本発明の実施形態に係る機構シミュレータは、このようなオペレーティングシステム上で動作するアプリケーションソフトウェアとして実装することができる。

#### 【0052】

次に、ハイブリッドモデル前処理部801における処理について説明する。ハイブリッドモデル記述102は、まずハイブリッドモデル前処理部801のモデル方程式制御情報解析部811において処理され、モデル方程式登録プログラム806、モデル方程式制御プログラム807が生成される。また、ハイブリッドモデルシミュレーション実行部102を構成するソフトウェアモジュールとして、モデル方程式の登録を行うための関数及び連続系方程式を切り替えるための関数がAPI(Application Program Interface)関数として提供される。モデル方程式登録プログラム806およびモデル方程式制御プログラム807は、該当する上記API関数を呼び出す記述を、入力されたハイブリッドモデル記述102に沿って適切に組み合わせたプログラムである。この観点から考えると、ハイブリッドモデル前処理部801は、入力をハイブリッドモデル記述102とし、出力を例えばC言語のAPI関数呼び出しの記述を含むCプログラム(ソース)とするような、一種のコンパイラと考えることもできる。このようなモデル方程式登録プログラム806とモデル方程式制御プログラム807は、さらにC言語などのコンパイラによりコンパイルされ、例えば実行時に動的にリンク可能なライブラリが生成される。ハイブリッドモデルシミュレーション実行部802は、シミュレーション実行にあたって、生成された動的リンクライブラリがリンクされ、入力ハイブリッドモデルを忠実に再現するシミュレーションプログラムが完成し、実行可能になる。その実行時には、まず方程式構文解析部812を起動するAPI関数が呼ばれ、その後に連続系方程式切り替えのAPI関数群が実行されて連続系シミュレーションが遂行される。

#### 【0053】

ハイブリッドモデルシミュレーション部802のアプリケーションインターフェースを

構成する具体的なソフトウェアモジュールの仕様などは様々考えられるが、ここでは説明の都合上、以下の3つのAPI関数が最低定義されているとする。なお、プログラミング言語はC言語とする。

【数7】

```
int XXX_AddEqnData(char*eqn,int*err)
int XXX_ActivateEqn(int eqnid)
int XXX_DeActivateEqn(int eqnid)
```

【0054】

1つ目のAPI関数XXX\_AddEqnDataは、1つの連続系方程式を表す文字列のポインタを引数に指定する。XXX\_AddEqnDataは、この連続系方程式を構文解析し、連続系方程式の記述をシミュレーション実行可能なデータ構造（内部データ表現）に変換し、かかる内部データ表現を方程式データ記憶部814に登録する処理を行う。なお、この連続系方程式には、ユニークなID番号が割り当てられる。

【0055】

たとえば「 $ab/\cos(a-(c+b))-3c$ 」という式が与えられたと仮定すると、上記内部データ表現として図9のような木構造を生成する。この木構造において、例えば参照数字61は線形多項式の親ノード（節）、62は掛け算のノード、63は割り算のノード、64は外部関数（四則演算以外の意）のノード、65は線形多項式を構成する各項のノードを表している。本例において、木構造の葉に相当するものはすべて変数（ $a$ 、 $b$ 、 $c$ ）であり、これらに実数の係数が加わって線形式となる。線形式は $\cos$ などの外部関数の引数になったり、掛け算や割り算の対象となる。変数には、別途、値が確定しているかどうかのフラグが設けられており、またこのような木構造のデータに基づいて該変数の現在の値が保持される。木構造のすべての葉の値（すなわち変数の値）が確定していれば、式の値を計算することができる。方程式データ記憶部114では、式の値の計算などを高速に行うことができるように、予め内部のデータ構造をつなぎ合わせて木構造を構成してある。

【0056】

上記処理において何らかのエラーが発生した場合には、`err`にエラーコードがセットされる。正常に処理が終了した場合は、登録された方程式のID番号を返り値とする。

【0057】

2番目のAPI関数XXX\_ActivateEqnは、引数に指定された方程式のID番号に相当する方程式を有効にする。もし、すでに有効となっている方程式が指定されている場合には何もしない。返り値はエラーコードである。

3番目のAPI関数XXX\_DeActivateEqnは、XXX\_ActivateEqnとは逆に、引数に指定された方程式のID番号に相当する方程式を無効にする。すでに無効となっている方程式が指定された場合には何もしない。

【0058】

モデル方程式制御情報解析部811は、まずXXX\_AddEqnDataを必要な方程式について順に呼ぶ関数（InitEqnData）を生成する。これがモデル方程式登録プログラム806に相当する。

【0059】

また、モデル方程式制御情報解析部811は、シミュレーション実行の際に、時間が $\Delta t$ 進むごとに条件のチェックおよび方程式の変更（入れ替え）を行う関数（ChangeEqn）も生成する。これはモデル方程式制御プログラム807に相当する。上記したようなハイブリッドモデル前処理部801における処理により、例えば、図7に示したハイブリッドモデル記述について、以下のようなC言語のソースプログラムが自動生成される。

## 【数 8】

```

static char eqn1[] = "f=mx";
static char eqn2[] = "f=mx";
static int eqn1id;
static int eqn2id;
int InitEqnData()
{
    int err;
    eqn1id = XXX_AddEqnData(eqn1,&err);
    if(err!=0)return err;
    eqn2id = XXX_AddEqnData(eqn2,&err);
    if(err!=0)return err;
    return 0;
}
int ChangeEqn()
{
    int err;
    BOOL GetEvent(char*eventname);
    If(GetEvent(Left)){
        Err = XXX_ActivateEqn(eqn1id);
        if(err!=0)return err;
        XXX_DeActivateEqn(eqn2id);
        if(err!=0)return err;
    }
    if(GetEvent(Right)){
        XXX_ActivateEqn(eqn2id);
        if(err!=0)return err;
        XXX_DeActivateEqn(eqn1id);
        if(err!=0)return err;
    }
}

```

## 【0 0 6 0】

なお、GetEventは、名前で指定されたイベントが生起しているかどうかをチェックする関数である。

## 【0 0 6 1】

以上のプログラムは、上述したようにC言語コンパイラによってコンパイルされ、さらに動的リンクライブラリの形式に整えられ、実行時にリンクされる。

## 【0 0 6 2】

なお、本実施形態では、プログラム言語としてC言語を用いた例について説明したが、本発明はこれに限定されるものではなく、例えばC P P (C++) 言語、S p e c C言語等の他のプログラム言語を用いてもよい。

## 【0 0 6 3】

次に、シミュレーションの実行について説明する。シミュレーション実行時においては、ハイブリッドモデルシミュレーション実行部 8 0 2 が起動され、図 2 に示す機構制御ソフトウェアのシミュレータ 1 0 8 などから得られる制御信号 1 0 9 を受信しながら、連続系方程式の値を計算することでシミュレーション実行が行われる。このとき、連続系方程式切り替え部 8 1 5 は、上記したモデル方程式制御プログラム 8 0 7 に基づいて、連続系方程式の切り替えを有効・無効のフラグを用いて実行する。図 4 の状態では、図 7 の運動方程式 e q 1 は無効であり、運動方程式 e q 2 が有効になっている。ここで、L e f t のイベントが発生した図 3 の状況においては、図 7 の運動方程式 e q 1 を有効にし、運動方程式 e q 2 を無効にするようフラグを操作する。これら有効・無効のフラグは方程式データ記憶部 8 1 4 に記憶される方程式それぞれの属性データとして管理される。



## 【0064】

連続系シミュレーション部803は、方程式データ記憶部814を参照し、同記憶部814に木構造の形式で格納されている連続系方程式の内部データ表現を演算対象として、時間ステップづつ数値積分を実行する。シミュレーションは、常微分方程式及び代数多項式の連立からなる非線形連立方程式についての初期値問題である。このため、例えば図4に示される初期状態が与えられている。具体的には、例えば一般によく使われているルンゲクッタアルゴリズムを用いて変数の値を計算する。

## 【0065】

必要なデータは機構シミュレータから出力を行い、さらに連続系方程式切り替え部815の処理に戻り、上記の処理を繰り返すことにより必要な時間のシミュレーションを実行する。シミュレーション結果は、変数値時間履歴記憶部805に保存され、シミュレーション終了後の分析などに利用される。

## 【0066】

【数1】－【数6】であらわされるようなハイブリッドモデル言語と、それを用いた図16のようなシミュレーションモデルをダイナミクスシミュレーション部101として利用し、キネマティクスシミュレーション部103と連携する場合を考える。図16のDCMLモデルを用いる場合には、図13に例示されている対応関係において、変数名として、`paper.val_`や`motor1.val_`などを指定することになる。これらは、設計者によって追加されるべきプログラムに含まれているので比較的指定しやすい。これに対し、3次元の機構モデルを動作させるために関係を付けなければならない変数でありながらダイナミクスモデルのなかでは、予め定義されたモデルにしか出現しない変数などについては、図13のような対応関係をモデル作成時に付けるのが困難である。

## 【0067】

たとえば、両軸のモータがあり、片方の軸は駆動用で、反対側の軸には冷却用のファンが直結しているような場合を考える。図16のようなモデルにおいて、設計者が追加するプログラムでは、ファンの回転角について考える必要はない。しかし、シミュレーションの際には、3次元キネマティクスシミュレーションによってファンの回転の様子をグラフィックス画面において確認したい場合には、ファンの軸を示す機構要素に対して、モータのクラス定義のみに現れるファン軸の変数名を指定する必要がある。しかし、あらかじめ定義されたクラスを多数読み込んで複雑なモデルを構築する場合には、キネマティクスシミュレーション用の要素とダイナミクスシミュレーション用の変数の両者を即座に選択して対応付けることが難しい。そこで本実施形態においては、モデルを読み込んだ際に、関係を付けられる要素を列挙し、可能性のあるもののみをユーザに提示して選択させるような構成にする。

## 【0068】

まず、キネマティクスモデルにおける3次元機構モデルの機構要素を列挙する。図12の例では、`Slid1`と`Joint1`の移動量のみが機構要素と成り得るので、これを画面に表示し、ユーザに選ばせる。図17は、このようなユーザ入力画面10（入力ダイアログ）の一例を示している。ここでは、機構要素はすでに入力済みであり、選択された機構要素の名称101が表示されている。選択可能要素ボタン102を押すことにより、選択可能な機構要素を選択することができる。

## 【0069】

一方、ダイナミクスモデル変数については、事前登録されたモデルから変数を選ぶためのボタン103とそれ以外の変数を選ぶためのボタン104が用意されている。事前登録されたモデルから変数を選ぶためのボタン103は、読み込まれたクラスに関係する変数を選ぶためのボタンであり、それ以外の変数を選ぶためのボタン104はクラスのメンバでない変数を選ぶためのボタンである。それ以外の変数を選ぶためのボタン104を押した場合には、ポップアップメニューにそのような変数の一覧が表示され、その中から選択をする。

## 【0070】

事前登録されたモデルから変数を選ぶためのボタン 103 が押された場合には、クラスの一覧を表示し、その一覧のなかからユーザにクラスを選択させる。図 18 はその選択画面 20 を示している。図 16 のダイナミクスモデルの例では、Motor と Paper の 2 つのクラスがあるので、これらは共に表示されている。ここで、Motor が選択されると、さらに図 19 のように部品の実体を選択させるポップアップメニュー 201 が表示される。ここでは、Motor の実体（インスタンス）としては motor1 が 1 つだけ定義されている。これがユーザに選択されると、図 20 のようにパブリックな連続変数のみの一覧 2001 が表示される。ここで、val\_ が選択され、ユーザにより OK ボタン 2002 が押されると、図 21 のように motor1.val\_ (11) が指定される。

#### 【0071】

クラスの機能を利用するなどしてハイブリッドモデルのモデル部品を定義しておく場合には、設計者が作成するダイナミクスモデルの記述は簡便になる一方、キネマティクスモデルとの対応関係定義などでは変数を探すのが複雑かつ面倒になる。これに対し本発明の第 2 実施形態によれば、ハイブリッドモデルの構造に即した形でデータを階層的に扱い、それに合わせた形態でメニューを階層的に構成することで、容易にキネマティクスモデルの機構要素とダイナミクスモデルの変数の対応関係定義が行えるようになる。

#### 【図面の簡単な説明】

#### 【0072】

【図 1】本発明の第 1 実施形態に係わる機構シミュレータの概略構成を示すブロック図

【図 2】本発明の第 1 実施形態に係わる機構シミュレータと機構制御ソフトウェアとが連携した状態を示すブロック図

【図 3】本発明の第 1 実施形態に係わる機構シミュレーションの処理手順を示すフローチャート

【図 4】本発明の第 1 実施形態に係わり、ハイブリッドモデル記述を説明するための具体例に係るシリンダ装置のある状態を示す図

【図 5】本発明の第 1 実施形態に係わり、ハイブリッドモデル記述を説明するための具体例に係るシリンダ装置の別の状態を示す図

【図 6】本発明の第 1 実施形態に係わり、ハイブリッドモデル記述を説明するための具体例に係るシリンダ装置の状態遷移を示す図

【図 7】本発明の第 1 実施形態に係わり、ハイブリッドモデル記述の内容を示す図

【図 8】本発明の第 2 実施形態に係わり、ダイナミクスシミュレーション部の一実施形態を示すブロック図

【図 9】本発明の第 2 実施形態に係わり、1 つの連続系方程式を構文解析した結果得られる内部データ表現の説明図

【図 10】本発明の第 1 実施形態に係わり、3 次元機構モデルにおける構成要素に関する説明図

【図 11】本発明の第 1 実施形態に係わり、3 次元機構モデルで定義された関係が計算機メモリなどの記憶手段にどのような形式で記憶されるかをグラフの形式で表現した図

【図 12】本発明の第 1 実施形態に係わり、機構の動きを 3 次元グラフィックスのアニメーションにて表現した図

【図 13】本発明の第 1 実施形態に係わり、変数・機構要素対応テーブル説明図

【図 14】本発明の第 2 実施形態に係わり、不連続な変化を含まない単純な DCML プログラムの一例を示す図

【図 15】本発明の第 2 実施形態に係わり、クラス定義の例を示す図

【図 16】本発明の第 2 実施形態に係わり、紙搬送系モデルの一例を示す図

【図 17】本発明の第 2 実施形態に係わり、入力ダイアログを示す図

【図 18】本発明の第 2 実施形態に係わり、クラス選択画面を示す図

【図 19】本発明の第 2 実施形態に係わり、部品の実体選択画面を示す図

【図 20】 本発明の第 2 実施形態に係わり、連続変数の一覧選択画面を示す図

【図 21】 本発明の第 2 実施形態に係わり、連続変数の指定結果を示す図

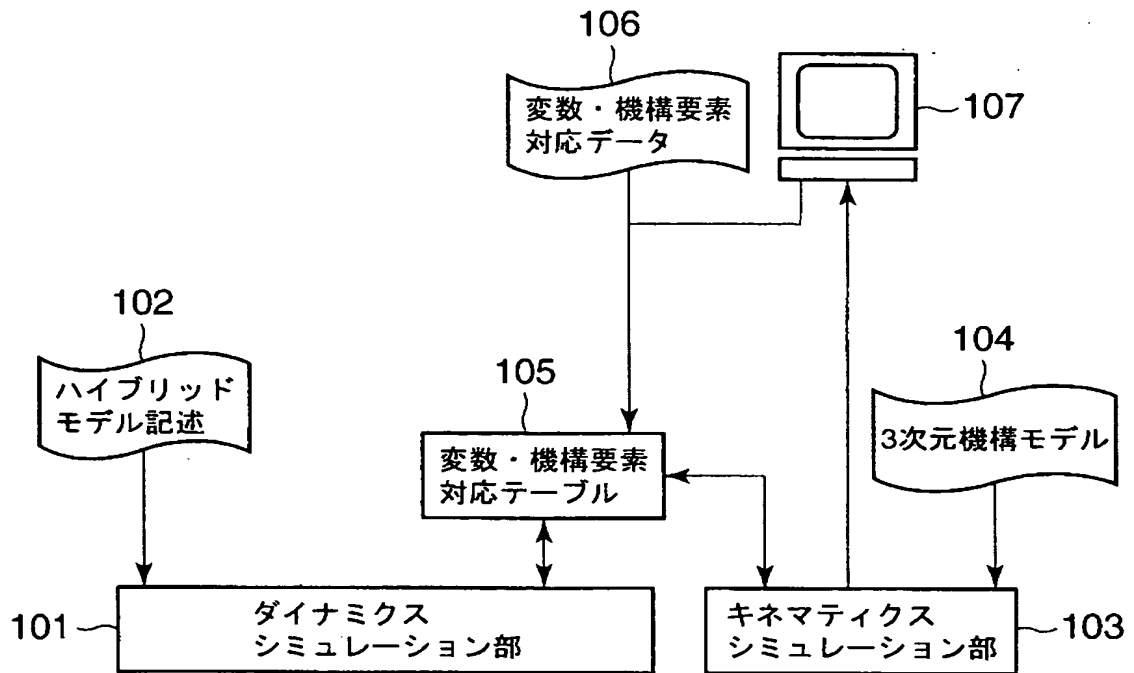
【符号の説明】

【0073】

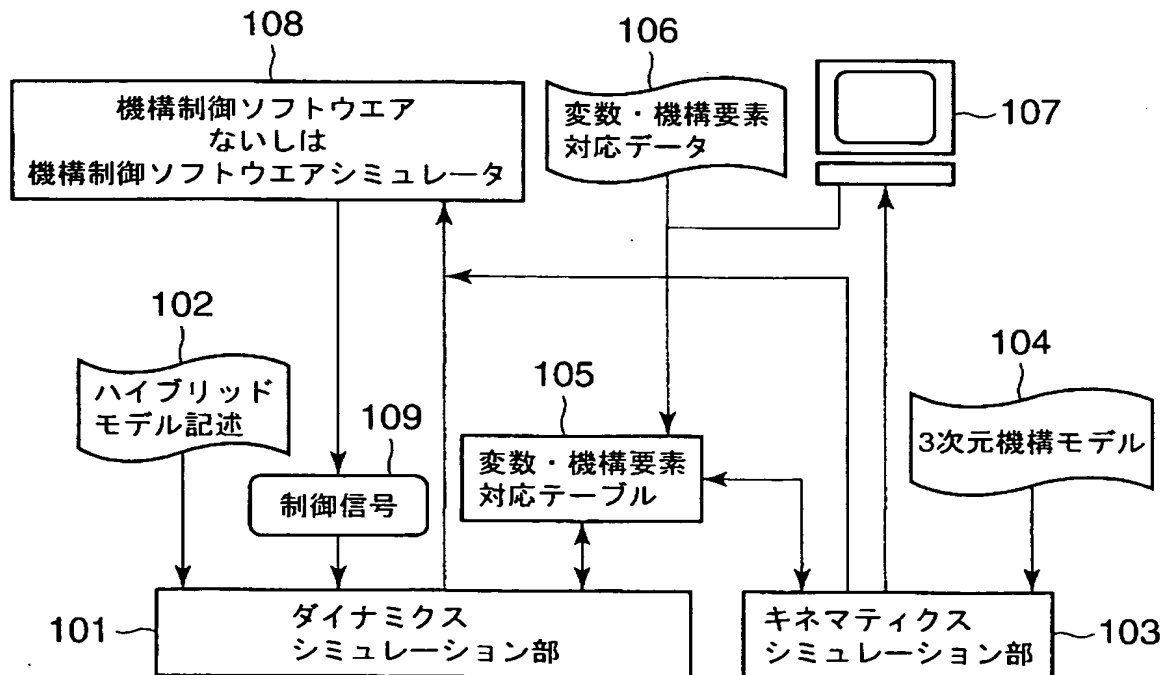
101…ダイナミクスシミュレーション部、102…ハイブリッドモデル記述、103…  
キネマティクスシミュレーション部、104…3次元機構モデル、105…変数・機構要  
素対応テーブル、106…変数・機構要素対応データ、107…コンピュータ、109…  
制御信号

【書類名】 図面

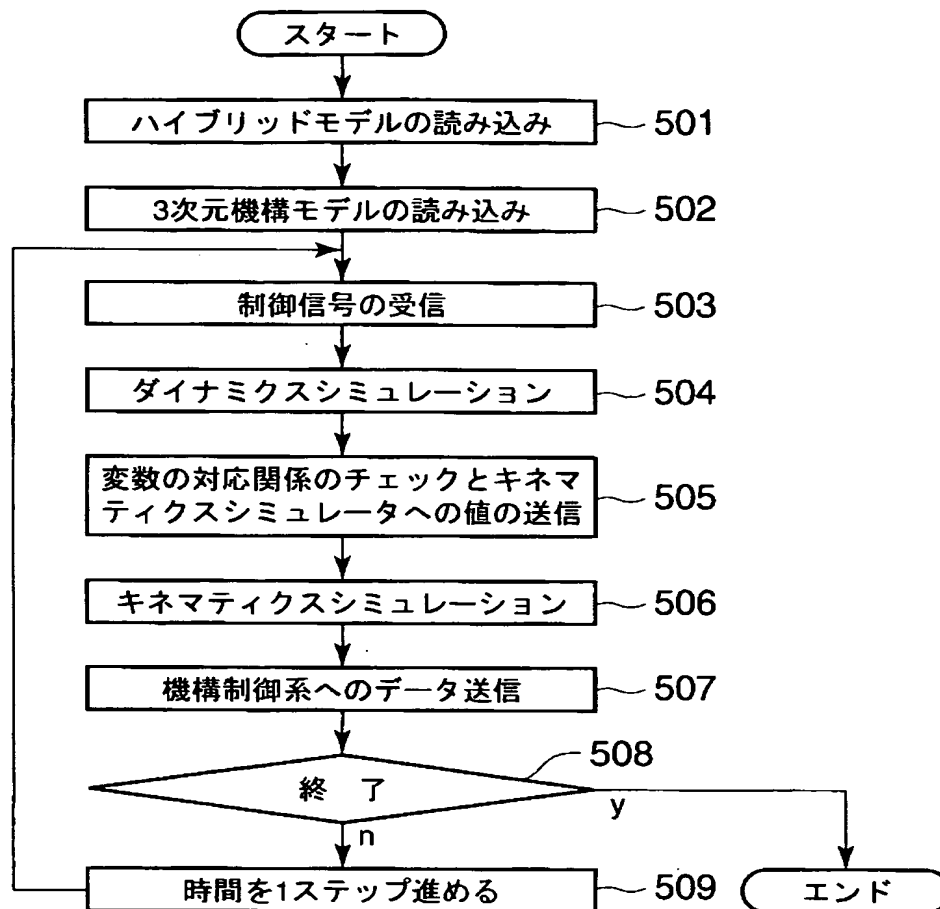
【図 1】



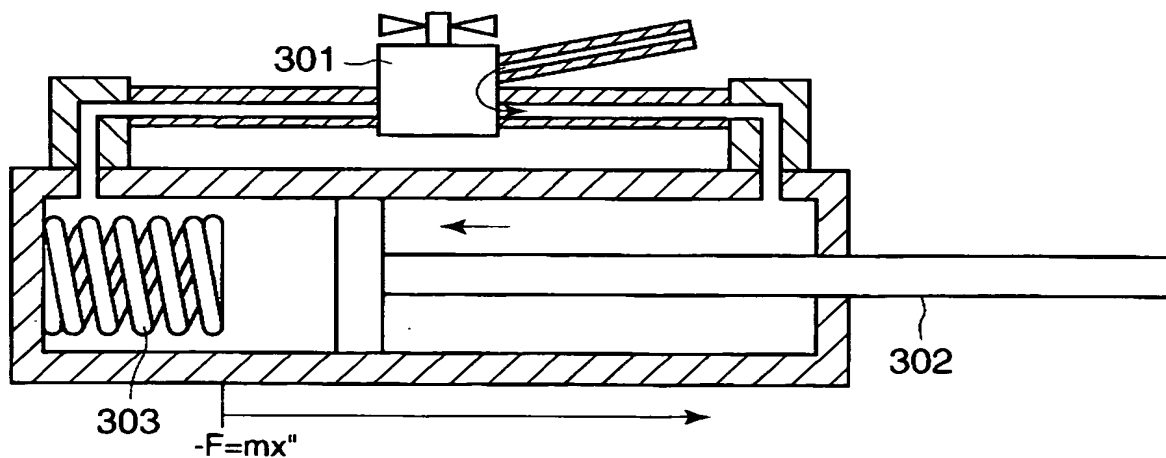
【図 2】



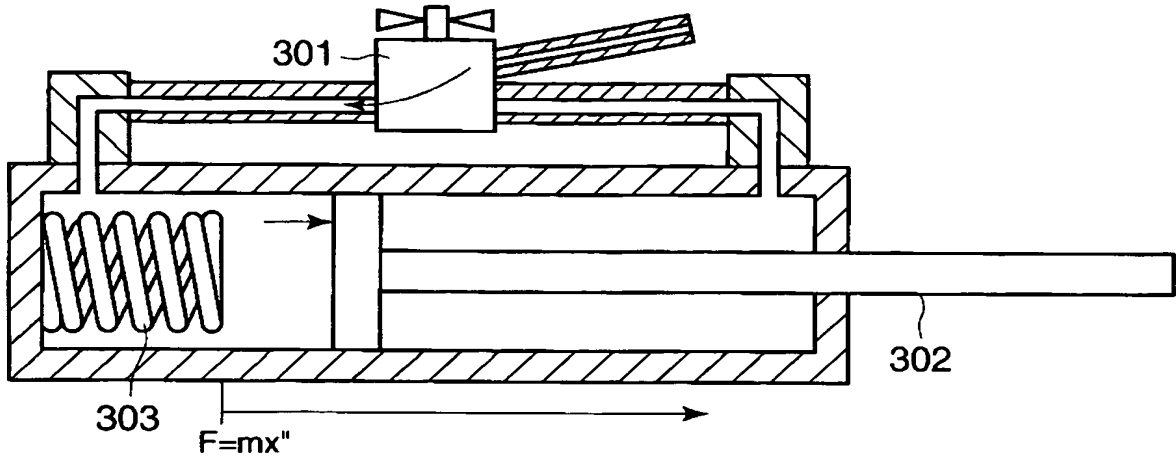
【図 3】



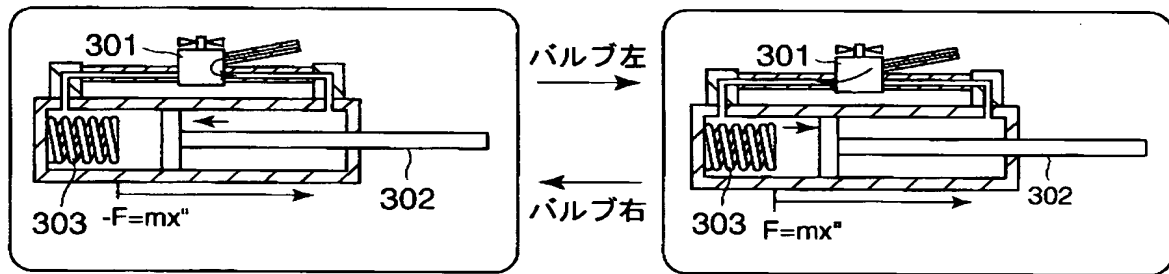
【図 4】



【図 5】



【図 6】



【図 7】

```

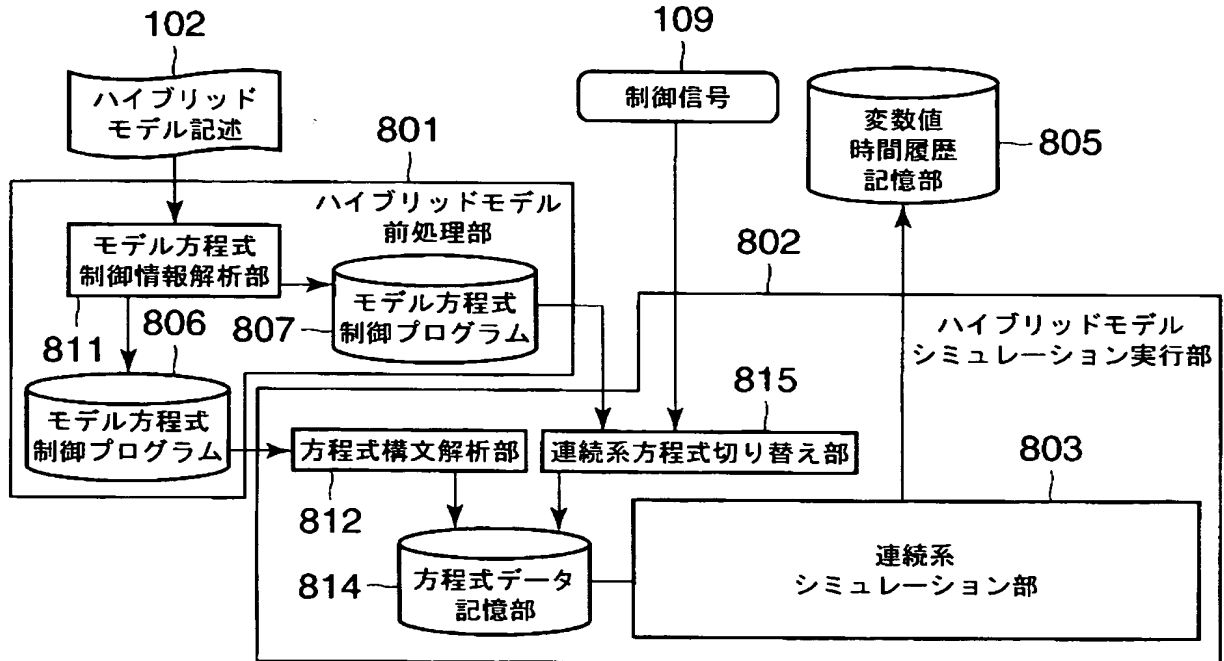
L1  #define m 1
L2  #define f 100
L3  Right ev1
L4  wait 50 do Left ev2
L5  always if Left then do always F=m*x'' watching Right,
L6  always if Right then do always -F=m*x'' watching Left,
L7  sample(x),
L8  x=0,x'=0,

```

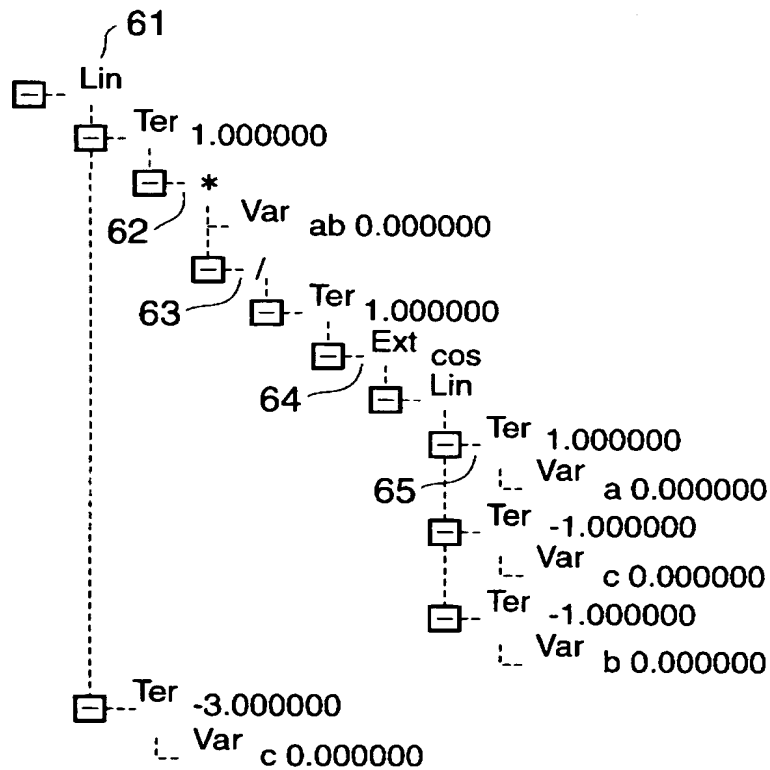
eq1

eq2

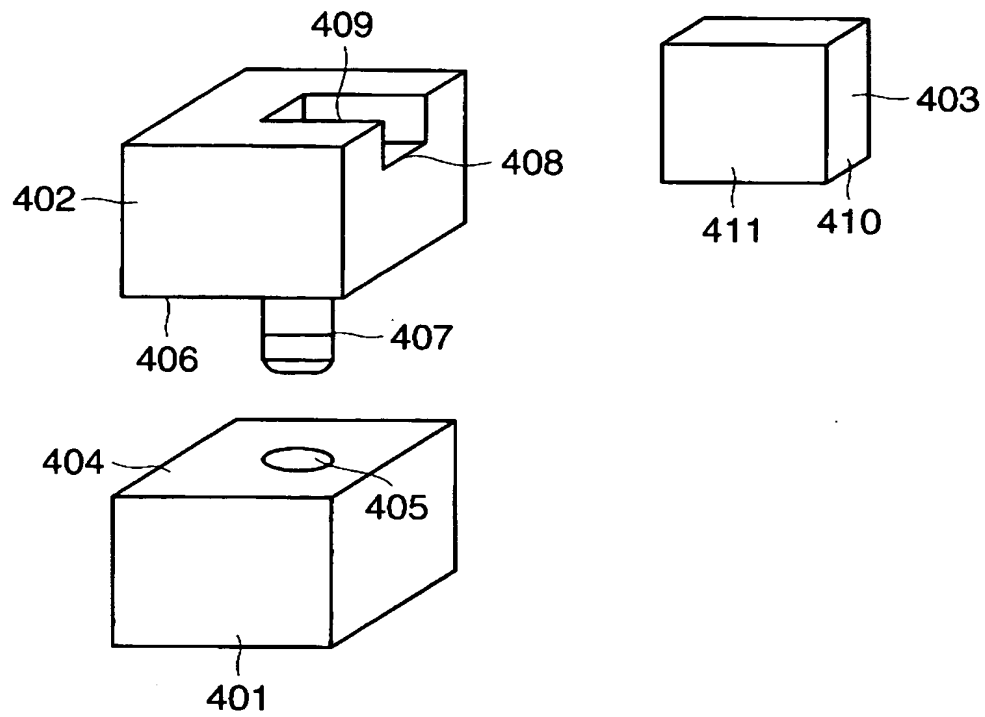
【図 8】



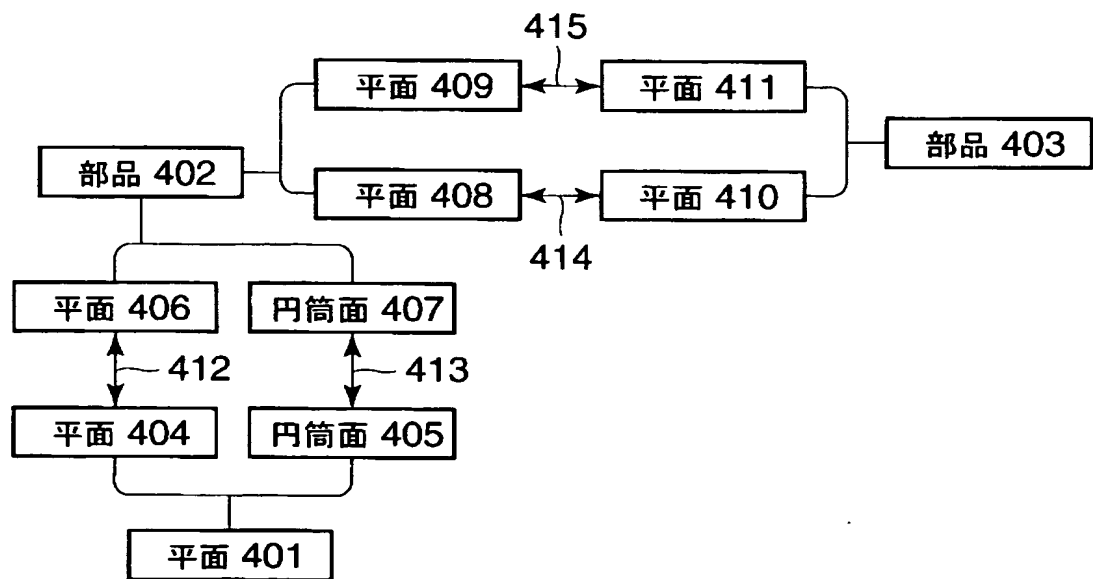
【図 9】



【図 10】

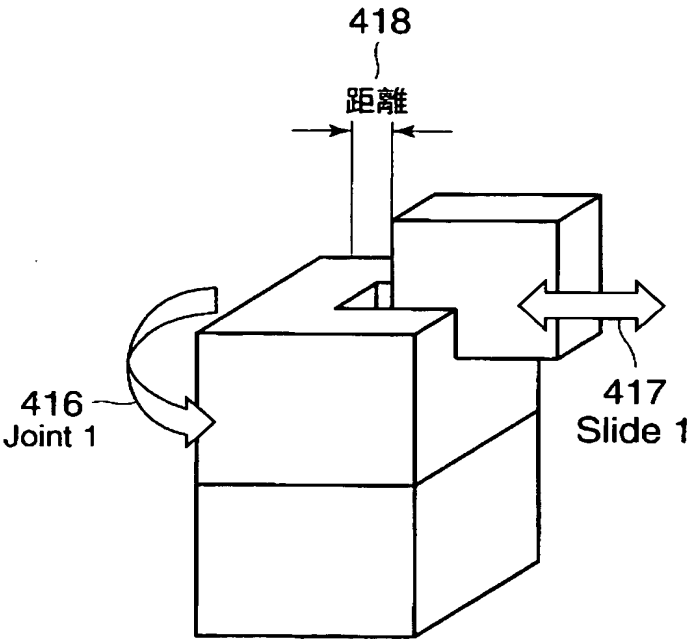


【図 11】





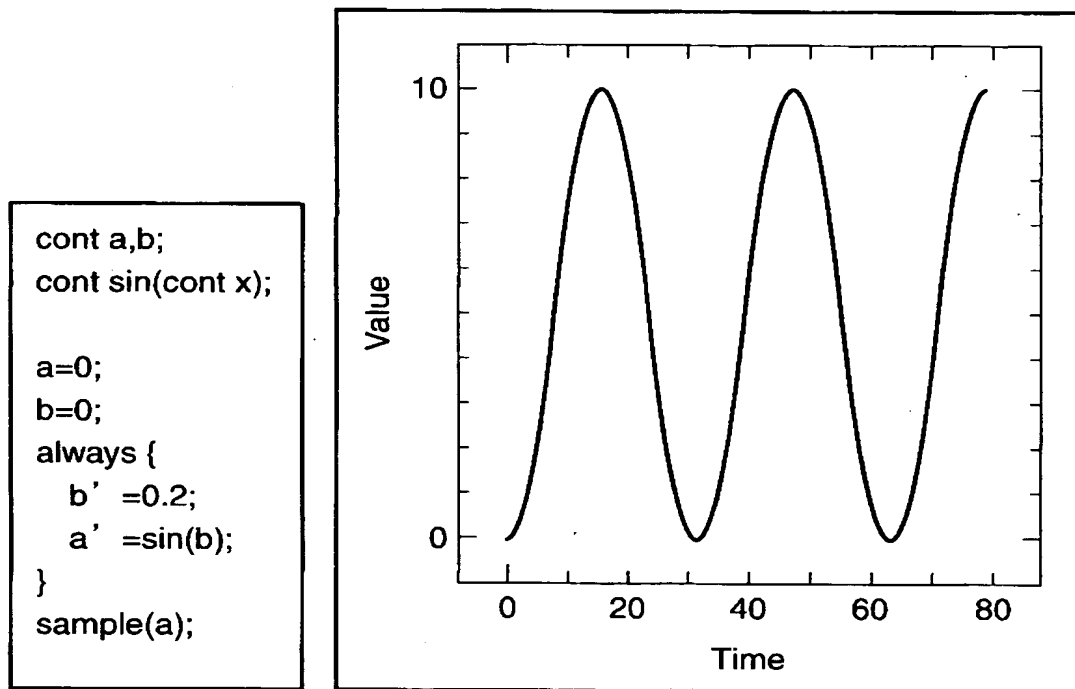
【図 1 2】



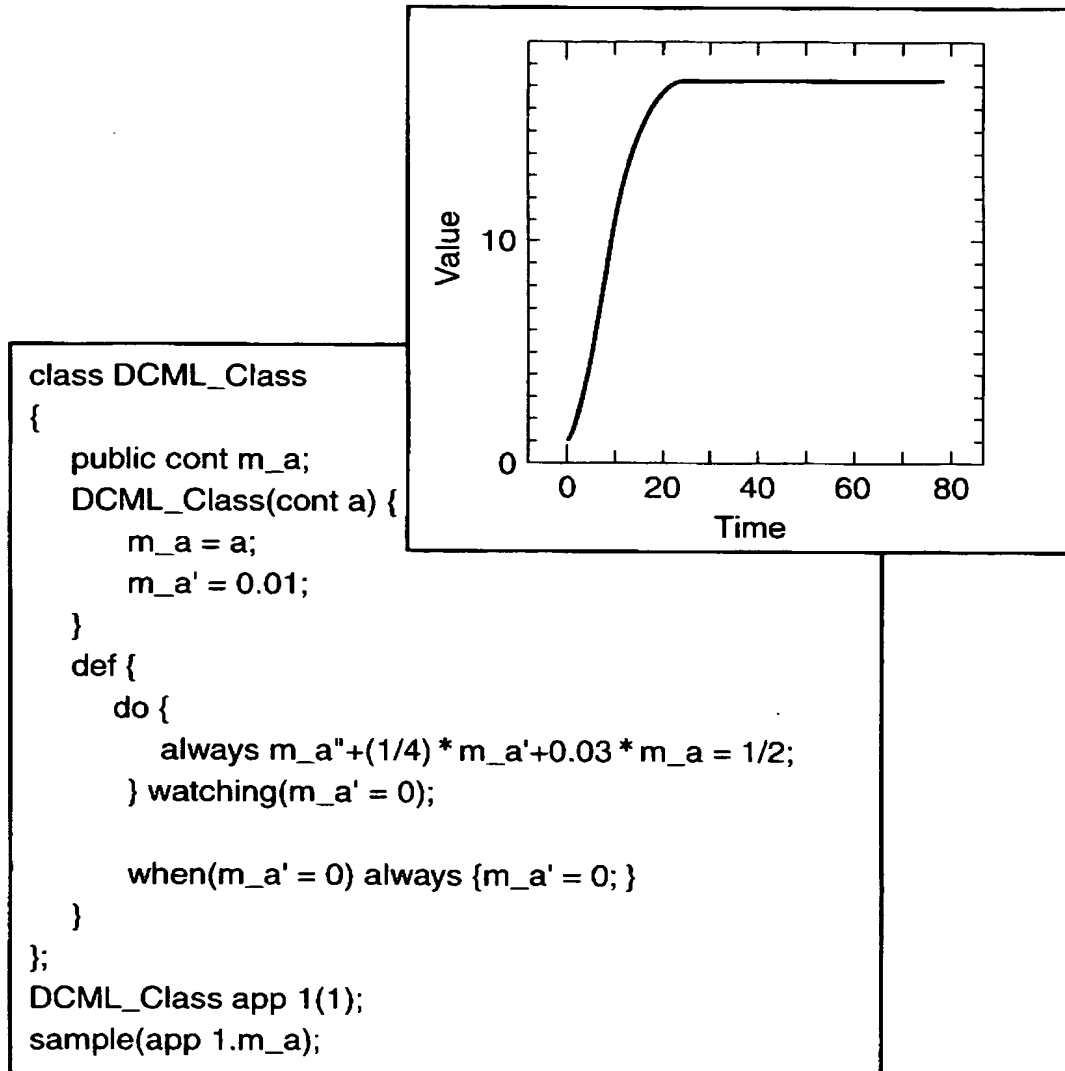
【図 1 3】

変 数	機構要素名
x	Slide 1
y	Joint 1

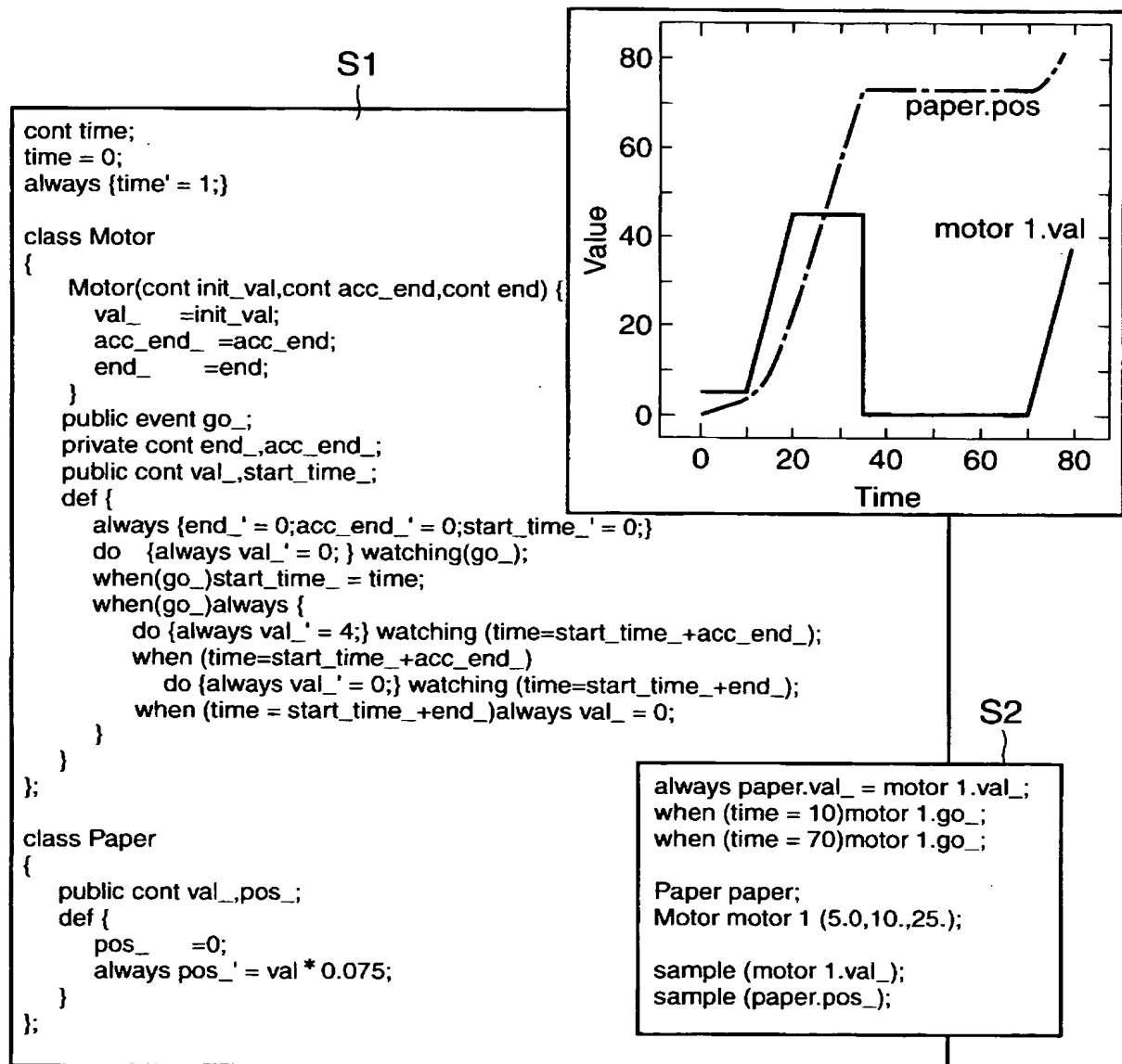
【図 14】



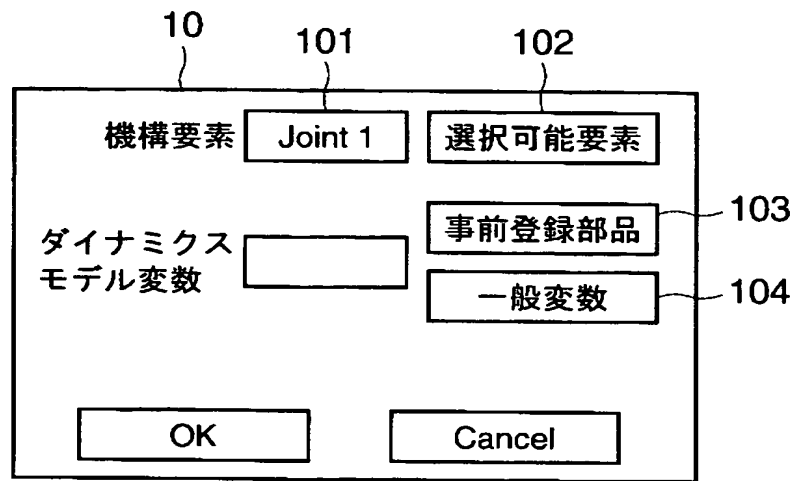
【図 15】



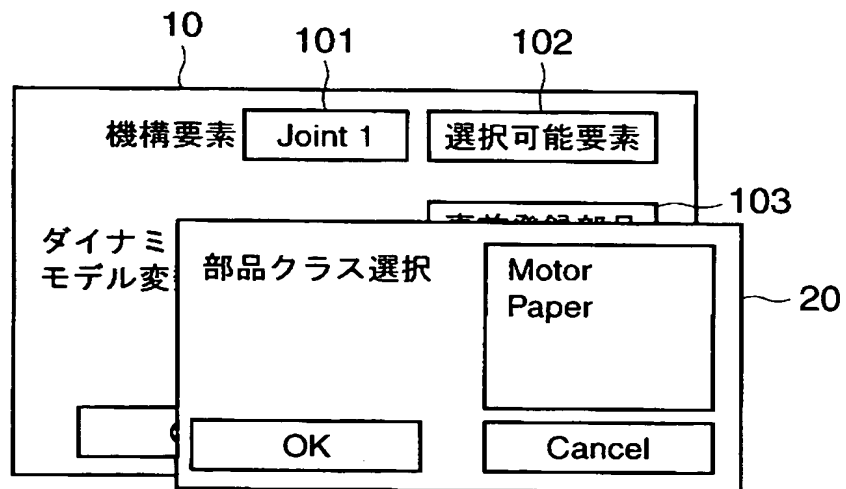
【図 16】



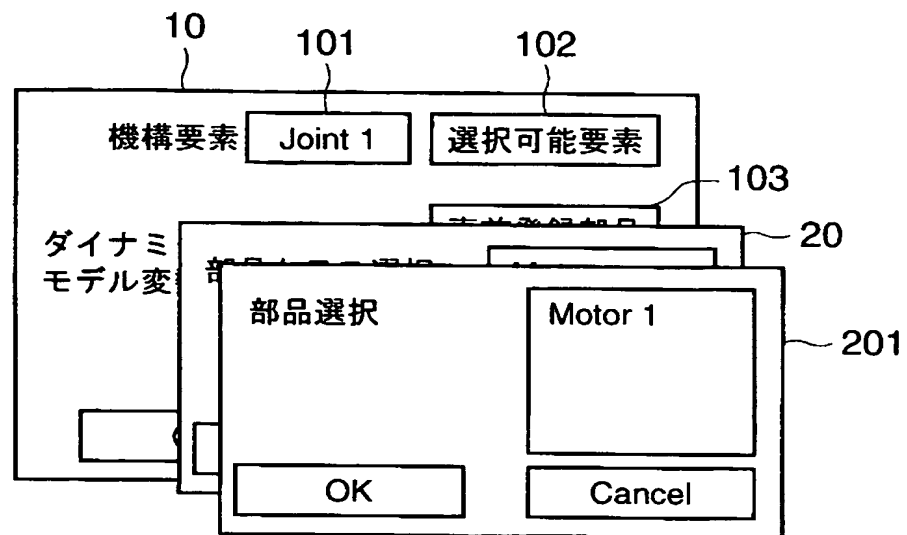
【図 17】



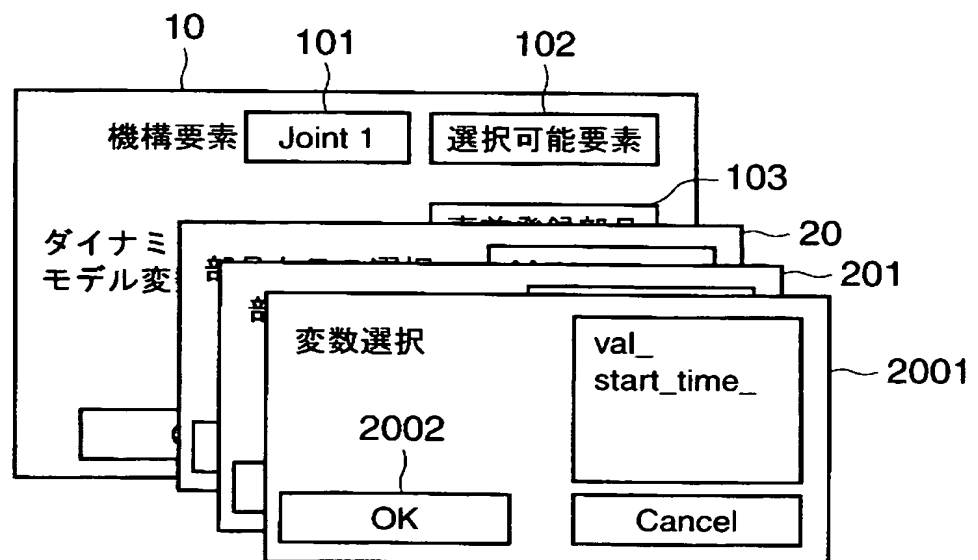
【図 18】



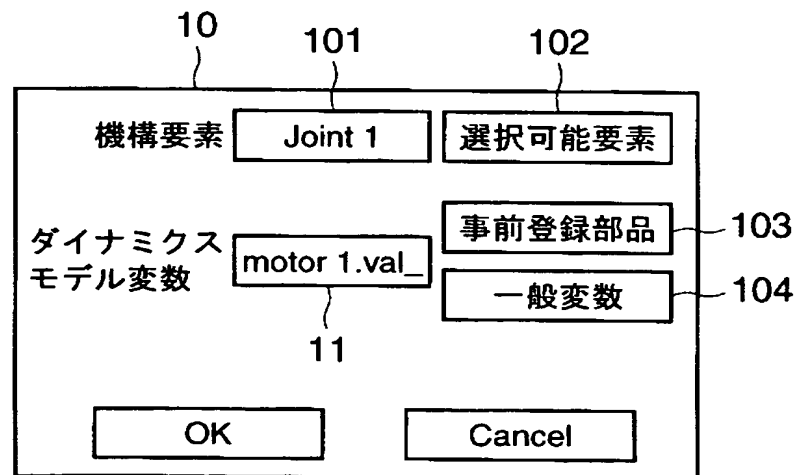
【図 19】



【図 20】



【図 21】



【書類名】 要約書

【要約】

【課題】 連続系方程式を用いて機構の挙動を時間軸に沿ってシミュレーションするダイナミクスシミュレーションと、3次元機構モデルを用いたキネマティックスシミュレーションとを併用する機構シミュレーション方法およびプログラムの提供。

【解決手段】 ダイナミクスシミュレーションと、キネマティックスシミュレーションとを併用する機構シミュレーション方法及びであって、連続系方程式における変数と3次元機構モデルの機構要素との対応関係を読み込むステッププログラムアップ501、502と、ダイナミクスシミュレーションを実行し連続系方程式の変数の値を求めるステップ504と、対応関係を参照し、変数の値に基づいて3次元機構モデルの機構要素に対するキネマティックスシミュレーションを実行するステップ506とを具備する。

【選択図】 図3



## 認定・付加情報

特許出願の番号	特願 2003-389710
受付番号	50301911980
書類名	特許願
担当官	第七担当上席 0096
作成日	平成 15 年 11 月 25 日

## &lt; 認定情報・付加情報 &gt;

## 【特許出願人】

【識別番号】	000003078
【住所又は居所】	東京都港区芝浦一丁目 1 番 1 号
【氏名又は名称】	株式会社東芝

## 【代理人】

申請人

【識別番号】	100058479
【住所又は居所】	東京都千代田区霞が関 3 丁目 7 番 2 号 鈴榮特許 綜合法律事務所内
【氏名又は名称】	鈴江 武彦

## 【選任した代理人】

【識別番号】	100088683
【住所又は居所】	東京都千代田区霞が関 3 丁目 7 番 2 号 鈴榮特許 綜合法律事務所内
【氏名又は名称】	中村 誠

## 【選任した代理人】

【識別番号】	100108855
【住所又は居所】	東京都千代田区霞が関 3 丁目 7 番 2 号 鈴榮特許 綜合法律事務所内
【氏名又は名称】	蔵田 昌俊

## 【選任した代理人】

【識別番号】	100084618
【住所又は居所】	東京都千代田区霞が関 3 丁目 7 番 2 号 鈴榮特許 綜合法律事務所内
【氏名又は名称】	村松 貞男

## 【選任した代理人】

【識別番号】	100092196
【住所又は居所】	東京都千代田区霞が関 3 丁目 7 番 2 号 鈴榮特許 綜合法律事務所内
【氏名又は名称】	橋本 良郎

【選任した代理人】

【識別番号】 100091351

【住所又は居所】 東京都千代田区霞が関 3 丁目 7 番 2 号 鈴榮特許  
綜合法律事務所内

【氏名又は名称】 河野 哲

特願 2 0 0 3 - 3 8 9 7 1 0

出 願 人 履 歷 情 報

識別番号

[ 0 0 0 0 0 3 0 7 8 ]

1. 変更年月日

2 0 0 1 年 7 月 2 日

[変更理由]

住所変更

住 所

東京都港区芝浦一丁目 1 番 1 号

氏 名

株式会社東芝